


SOLE INVENTOR

P17593

"EXPRESS MAIL" mailing label No.  
EV323779199US

Date of Deposit: December 16, 2003

I hereby certify that this paper (or fee) is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 CFR §1.10 on the date indicated above and is addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450



Richard Zimmermann

APPLICATION FOR  
UNITED STATES LETTERS PATENT  
  
SPECIFICATION

TO ALL WHOM IT MAY CONCERN:

Be it known that I, **Murthi Nanja**, a citizen of the United States of America, residing at 14314 NW Spruceridge Lane, Portland, Oregon 97229, has invented a new and useful **PERFORMANCE MONITORING BASED DYNAMIC VOLTAGE AND FREQUENCY SCALING**, of which the following is a specification.

**PERFORMANCE MONITORING BASED DYNAMIC VOLTAGE AND  
FREQUENCY SCALING**

**Field of the Disclosure**

**[0001]** The disclosure generally relates to voltage and frequency scaling in processor applications and, more particularly, to techniques for voltage and frequency scaling that use a performance monitor.

**Background of the Related Art**

**[0002]** Power dissipation and energy consumption are key parameters in designing embedded systems, such as cellular telephones, personal digital assistants (PDA), and multi-feature smart devices. These parameters are not only important for increasing battery life, they are important for reducing integrated device packaging costs and device cooling costs.

**[0003]** Reduced instruction set computing (RISC) architectures, such as million instructions per second (MIPS) machines and Advanced RISC machines (ARM), enjoy great popularity in the embedded systems market and, as a result, benefit from power and energy consumption management. The Intel XScale™ core architecture, available from Intel Corporation of Santa Clara, CA, is an example implementation of the ARM architecture. XScale™-based processors are used in the Intel Personal Internet Client Architecture (PCA) processors (e.g., PXA250, PXA261, and PXA262) employed in vast numbers of embedded systems.

**[0004]** Power and energy management in processor architectures can be done at several levels within an embedded system hierarchy, including, the integrated circuit component level, the platform level, the operating system level, the managed runtime environment level, the application level, and the

user level. In most systems, power and energy management is performed by the operating system. Some operating systems, for example, use dynamic frequency and voltage scaling techniques.

**[0005]** Frequency scaling is a technique where the processor clock is adjusted by a multiple of the maximum clock frequency or the memory bus frequency. This permits the processor to consume less power, but does so at the expense of reduced performance. Frequency scaled processors can have any number of discrete operating frequency points. For example, the Intel PXA250 processor supports the following operating frequencies in run mode: 99.5 MHz at 0.85V; 118, 132.7, 147.5, 165.9, and 199.1 MHz at 1.0V; 235.9, 265.4, and 294.9 MHz at 1.1V; and 381.9 MHz at 1.3V.

**[0006]** Dynamic voltage scaling reduces the power consumed by a processor by lowering its operating voltage. A reduction in operating voltage also requires a proportional reduction in frequency. Voltage scaling is preferred because the dynamic power consumed by a CMOS processor is directly proportional to the square of the operating voltage:  $P(\text{power}) = C(\text{capacitance}) * V^2 (\text{voltage}^2) * f(\text{frequency})$ , where C is the average switching capacitance loading of the processor, f is the operating frequency, and V is the supply voltage. Power consumption can be minimized by reducing C, f, or V. Reducing voltage will have a quadratic effect on power. By varying the voltage and the frequency, it is possible to obtain more than a quadratic reduction in power dissipation. Scaling down frequency without scaling voltage is typically not done, however, since the power savings is offset by the increase in execution time, yielding no reduction in the total amount of energy consumed.

**[0007]** The dynamic voltage and frequency scaling techniques executed in operating systems today use interval-based schedulers. Although prevalent, interval-based schedulers have numerous shortcomings.

**[0008]** One shortcoming is that interval-based schedulers merely predict a future workload. They use algorithms based on uniform-length intervals (in the range of 30 to 100 ms) to monitor the processor utilization of a previous interval. That historical data is then used to set the voltage level for the next interval. Although the interval-based scheduling algorithm is simple and easy to implement, it often predicts the future workload incorrectly, especially when an application's workload exhibits large variability, e.g., cycle to cycle variability. Interval-based schedulers are unable to scale the voltage and frequency of the processor at runtime based on actual usage patterns of the executing application.

**[0009]** A second shortcoming is that interval-based schedulers predict usage based on a processor utilization factor unrelated to future workload. And there is no standardization by which that utilization factor can be made to accurately predict future workload.

### **Brief Description of the Drawings**

**[0010]** FIG. 1 illustrates a block diagram of a system including a central processing unit (CPU) architecture having a performance monitor unit (PMU).

**[0011]** FIG. 2 illustrates the performance monitor unit of FIG. 1 in greater detail.

**[0012]** FIG. 3 illustrates a power management architecture that may be implemented on the system of FIG. 1

**[0013]** FIG. 4 illustrates an example implementation of the power management architecture for an example operating system.

### **Detailed Description of an Example**

**[0014]** Apparatuses and techniques are described that allow dynamic voltage and frequency scaling, for example, using an Intel XScale™ core architecture with a performance monitor. In these examples, voltage and frequency scheduling policy may be optimized using a voltage scheduler component executing in an underlying operating system. The techniques may reduce power dissipation and energy consumption of end user applications implemented either in native code or in managed code such as Java and C#. The techniques may be used in embedded devices, such as portable personal computers, PDAs, cellular telephones, smart phones, and other portable electronic devices. Although the techniques are described with reference to such embedded devices, persons of ordinary skill in the art will appreciate that the techniques may be used in other processor environments, including non-embedded devices. Further still, while the described techniques affect operating voltage and operating frequency, information from a performance monitor may be used to affect other power management variables.

**[0015]** FIG. 1 illustrates an example computer system 100 that includes a CPU unit 102 coupled to a Level 2 cache 104, via a CPU bus 106. The cache 104 is coupled to a RAM 108 and a read-only memory (ROM) 110, via a memory bus 112. In the illustrated example, the memory bus 112 is coupled to a system bus 114. Alternatively, the memory bus 112 may be a system bus. Persons of ordinary skill in the art will appreciate that the illustrated configuration is by way of example only.

**[0016]** The CPU 102 may include a discrete arithmetic logic unit (ALU), registers, and control unit all connected together. Or, as shown, the CPU 102 may be an integrated microprocessor. The CPU 102 includes a Level 1 cache 116, which may include a data cache, an execution cache, and an instruction cache running at processor speeds.

**[0017]** The CPU 102 also includes a performance monitoring unit (PMU) 118 that may be on the CPU chip, as shown, or coupled thereto. Suitable microprocessors offering on-chip PMUs include Pentium® 4, Itanium®, and XScale™-based processors (all available from Intel Corporation), as well as the PowerPC® line of microprocessors available from IBM Corporation of White Plains, New York. The CPU 102 may be any processor or processor architecture (e.g., one with an off-chip PMU) capable of performance monitoring. And, although not shown, persons of ordinary skill in the art will recognize that the CPU architecture 102 may also include a memory management unit, branch target and write buffers, as well as support logic for debugging.

**[0018]** The system bus 114 is coupled to a network controller 120, a display unit controller 122, an input device 124, and a data storage/memory medium 126, e.g., a mass storage device. Examples of the various devices coupled to the bus 114 are known. In the illustrated example, the bus 114 is coupled to another bus 128 via a bus bridge 130.

**[0019]** The operating system operating within the processor architecture 102 may be one of a variety of systems, for example, one of the WINDOWS family of systems available from Microsoft Corporation of Redmond, Washington, such as WINDOWS 95, 98, 2000, ME, XP, or CE. Alternatively,

the operating system may be one of the UNIX\* family of systems, originally developed by Bell Labs (now Lucent Technologies Inc./Bell Labs Innovations) of Murray Hill, New Jersey and available from various sources. As a further alternative, the operating system may be an open-source system, such as the LINUX operating system. It will be recognized that still further alternative operating systems may be used.

**[0020]** FIG. 2 illustrates the PMU 118 in greater detail. The PMU 118 includes control logic 150, counters 152, and registers 154. The PMU 118 may be on-chip hardware that monitors discrete performance events during program execution. The counters 152 may include global time stamp counters and dedicated programmable event counters.

**[0021]** The events monitored may be identified by the event registers 154, which control the counters 152 to incrementally monitor desired events, such as runtime performance data indicative of thread level utilization of the CPU 102. Example performance events include instruction cache misses, data cache misses, instructions executed, branches executed, branch mis-predicts, instruction translation look up buffer (TLB) TLB misses, data TLB misses, stalls due to data dependency, data cache write-back, etc. Each register within the registers block 154 may control a number of counters within the counters block 152. By way of example only, 32 bit-counters and 32-bit or 64-bit registers may be used, respectively. Different processor architectures may support simultaneous monitoring of more than one performance event.

**[0022]** FIG. 3 shows an example system architecture 200 of a dynamic voltage and frequency scaling system. The system 200 includes a user mode and a kernel mode that runs an operating system environment 202. The

operating system 202 of the embedded platform serves as a software interface between end-user applications 204 and platform hardware 206. For example, the end-user applications 204 communicate with an abstraction of the platform hardware 206 provided by the operating system 202. All application programs run in the user mode, while the operating system components run in the kernel mode of the underlying processor.

**[0023]** The architecture 200 can be used to reduce the power dissipation and energy consumption of the underlying processor subsystem via dynamic voltage and frequency scaling. This can prolong the battery life of an embedded device. The load of the CPU 102 will change over time, depending on the application types and user interactions with the application. That is, the power dissipation and energy consumptions of the CPU 102 depend on the type and the number of instructions executed by the application program. The power and energy consumptions also depend on the memory reference patterns of the executing applications. By observing the number of instructions executed and the total number of memory references of the application at runtime, or other performance events, the architecture 200 can effectively adjust power management within the system 100, for example, via adjusting operating voltage and operating frequency.

**[0024]** The platform hardware 206 includes the CPU 102 which is capable of supporting dynamic clock frequency and voltage scaling. Example processors having dynamic clock frequency and voltage scaling capabilities are listed above. These processors are capable of operating at different frequencies and at different supply voltages, for example, via a programmable



voltage regulator that supplies appropriate voltage to the processor at different operating frequencies.

**[0025]** As would be understood by persons of ordinary skill in the art, the platform hardware 206 may include additional subsystems and these have been collectively labeled 208 in FIG. 3.

**[0026]** The operating system 202 includes a performance monitor component 210 coupled to the CPU 102 directly or indirectly and is able to capture performance monitored events and current processor frequency. The monitor 118 is a hardware component that is part of the processor 102, and the monitor 210 is a software component that makes use of the monitor 118 to get performance event data. The monitor 210 can directly interact with the monitor 118 or it can interact with monitor 118 through some intermediary software driver component. The monitor 118 provides the hardware mechanism to gather performance event data, while the monitor 210 presents code to gather the performance events. The performance monitor 210 observes the usage patterns of the executing application at its thread level at runtime. In other words, the performance monitor 210 collects execution characteristics or usage patterns of every application thread running on the system. For Intel XScale™ architecture for example, a set of execution characteristics can be obtained by using the PMU of the Intel XScale™ core, which provides two 32-bit performance counters that allow two performance events to be monitored simultaneously. Additional counters may be used to monitor more than two simultaneous events. The Intel XScale™ core also has a 32-bit clock counter which can be used in conjunction with the performance counters to measure the cycle count for the executing

application. The current Intel XScale™ core provides 14 different performance events that can be monitored. These include events such as those highlighted above.

**[0027]** Several execution characteristics of an application thread can be gathered using the performance monitor 210. Since the Intel XScale™ core supports monitoring only two events at a time, instruction counts and memory references may be monitored for thread-level utilization in that processor architecture. Other characteristics may be used, however, and the characteristics used may depend on the processor employed. Additionally, the performance monitor may monitor multiple application threads simultaneously.

**[0028]** With instruction counts and memory references, along with the cycle counts for the application, the performance monitor 210 may calculate two metrics: instructions per clock cycle and memory references per clock cycle for the application thread. The instructions per clock cycle metric indicates the sensitivity for performance loss due to reduction in operating frequency, while the memory reference per clock cycle metric indicates the degree of memory use by the application. These two metrics may be calculated from the performance events measured by the PMU, described by example above. These two metrics are examples only, however. It is possible to create additional metrics based on other performance events data.

**[0029]** The performance monitor 210 may also observe the current operating frequency of the processor at the application thread level. The performance monitor 210 provides both thread level execution characteristic metrics and the current processor clock frequency information to a

power/energy policy controller 212 at runtime. And, the power/energy policy controller 212 determines the desired operating voltage and operating frequency. The operating system 202 may configure a default power/energy policy, having a pre-determined fixed voltage and clock frequency for the platform for each application thread. During an application run, this default policy may be optimized based on observation data, such as the two thread-level characteristic metrics.

**[0030]** The power/energy policy controller 212 may access a voltage and frequency lookup table 214, for example, and compare the thread-level characteristic metrics (or metrics derived therefrom) to the values stored in the lookup table 214. The table 214 may be constructed by empirical analysis of appropriate processor workloads or micro-benchmarking programs to create different instructions per clock and memory references per clock metrics. The table 214 may provide a range of optimal processor clock frequencies and an optimal minimum voltage for each instructions-per-clock cycle metric and each memory-references-per-clock cycle metric to conserve power and energy with minimal impact on the application performance. Depending on the characteristic metrics of the application thread, the power/energy policy controller 212 may use the lookup table 214 to select the appropriate clock frequency and voltage for the subsequent thread execution. The table 214 may include multiple operating frequencies for a particular operating voltage. The power/energy policy controller 212 may select among these operating frequencies based on the current operating frequency, for example. The power/energy policy controller 212 may determine a new operating frequency, a new operating voltage, or both.

**[0031]** Once a decision has been made regarding the operating clock frequency and the operating voltage, the decision information is passed to a voltage and frequency scheduler 216 that adjusts the processor frequency and corresponding supply voltage on the fly prior to executing the application thread. Techniques for changing processor frequency and voltage supply are known. On an Intel XScale™-based processor, for example, changing the clock frequency is achieved by writing appropriate multiplier values into a configuration register in the CPU architecture. A programmable voltage regulator may be used to scale the voltage.

#### **Example Implementation**

**[0032]** The blocks 210 - 216 may be implemented as part of one or more operating system components. An example implementation is shown in FIG. 4 and may correspond to an example UNIX operating system implementation.

**[0033]** In the process 300, the architecture 200 is implemented via existing operating system kernel data structures 302, thread timer interrupt routine 304, context switch routine 306, and a thread scheduler component 308. The voltage and frequency lookup table 214 may be a part of the kernel data structures 302.

**[0034]** The timer interrupt routine 304 gathers or derives instruction counts and memory references (i.e., based on PMU monitored events) and cycle counts, and stores them in the kernel data structures 302. The context switch routine 306 also maintains these two-thread level PMU values, cycle counts, and the current operating frequency at the thread level. The thread scheduler 308 calculates the thread characteristics metrics (i.e., instructions per cycle and memory references per cycle), determining the desired operating

frequency and operating voltage levels for the given thread characteristics metrics via the frequency and voltage lookup tables. The scheduler 308 also performs the dynamic voltage and frequency scaling. Depending on the operating systems, the techniques may be implemented either as a stand-alone module or component of the operating system or as an extension to the existing kernel modules or components.

**[0035]** Although some of the techniques are described in the context of the Intel XScale™ architecture, the voltage and frequency scaling architecture are applicable to other architectures, such as the IA32 architectures, also available from Intel Corporation. The differences are mainly in the performance monitoring capabilities of the processor and in the voltage and frequency lookup tables for different processor architectures. Furthermore, the techniques described can be extended to include additional thread-level utilization metrics, e.g., with processor architectures capable of monitoring more than two performance events simultaneously.

**[0036]** Although certain apparatus and techniques constructed in accordance with the teachings of the invention have been described herein, the scope of coverage of this patent is not limited thereto. On the contrary, this patent covers all embodiments of the teachings of the invention fairly falling within the scope of the appended claims either literally or under the doctrine of equivalence.